Sensor Data Processing on a Reconfigurable Processor

Gregory Donohoe
Institute of Advanced Microelectronics
ECE/CAMBR
University of Idaho

Pen-Shu Yeh
NASA GSFC
Code 567

RDPP Project Team

Co-Investigators

Gregory Donohoe, CAMBR
Pen-Shu Yeh, NASA GSFC

Student Research Assistants

Enrique Coen & Mrinal Kochar, *Architecture & chip design*
David Buehler, *Software lead*
Jagdish Sabde, *Application development*
Ching Choy & Tu Le, *Hardware emulation*

Professional

Joe Hass, CAMBR, VLSI Lead
Dr. Jay Herman, NASA Science Advisor
Dr. William Barnes, NASA Science Advisor

Purpose

Demonstrate two challenge applications on the Reconfigurable Data Path Processor.

Highlight key RDPP Features

Reconfigurability

Data path parallelism

Dynamic data path selection

Overview

Background on the Reconfigurable Data Path Processor

Focal Plane Array Sensor Readout Correction

Fourier Transform Hyperspectral Imager Data Conversion

Results & Conclusions

What is the RDPP?

An Embedded data processor VLSI chip for spacecraft

- Targeted to rad-tolerant, 0.25µ CMOS process.
- Implements a reconfigurable, synchronous data pipeline.
- Run-time reconfigurable.
- Serves as co-processor to a host CPU.
- Off-loads data intensive, streaming tasks from host.

Suite of support software

- Application development, compiler, simulator, run-time.
- Integrated with existing software platforms.

Low-power, radiation-tolerant alternative to Field Programmable Gate Arrays for reconfigurable spacecraft data processing.

Reconfigurable Computing

Reconfigurable (Adaptive) Computing: Processors that rewire themselves “on the fly” to optimize their architectures for the task at hand.

“The flexibility of software with the performance of dedicated hardware.”

-- Dr. José Muñoz, DARPA Adaptive Computing Program

The RDPP is one of the very few processors designed specifically for reconfigurable (adapative) computing.

Reconfigurable in the News

Computing's Big Shift: Flexibility in the Chips

New York Times, June 16, 2003

"We're coming upon a sea change in the world of semiconductors. There are compelling advantages of reconfigurable chips in terms of performance and power consumption."

-- Nick Trendennick, Microprocessor Pioneer

RDPP Features

Incrementally Reconfigurable

Can address and configure individual elements

Supports partitioning of computational problem

Large problem spread over multiple configurations

Scalable

Tile multiple RDPPs to form a parallel computing fabric. Example: 2D image filter

Size	**No. RDPPs**	**Throughput**
3x3	1 RDPP	60 MSamples/sec
5x5	2 RDPPs	60 MSamples/sec
7x7	4 RDPPs	60 MSample/sec
⋮	⋮	⋮

RDPP Features (cont.)

High Throughput on Streaming Tasks

Massive data path parallelism

Intrinsically Radiation Tolerant

Fabricated in rad-tolerant CMOS technology

No three-module redundancy required

No configuration memory scrubbing

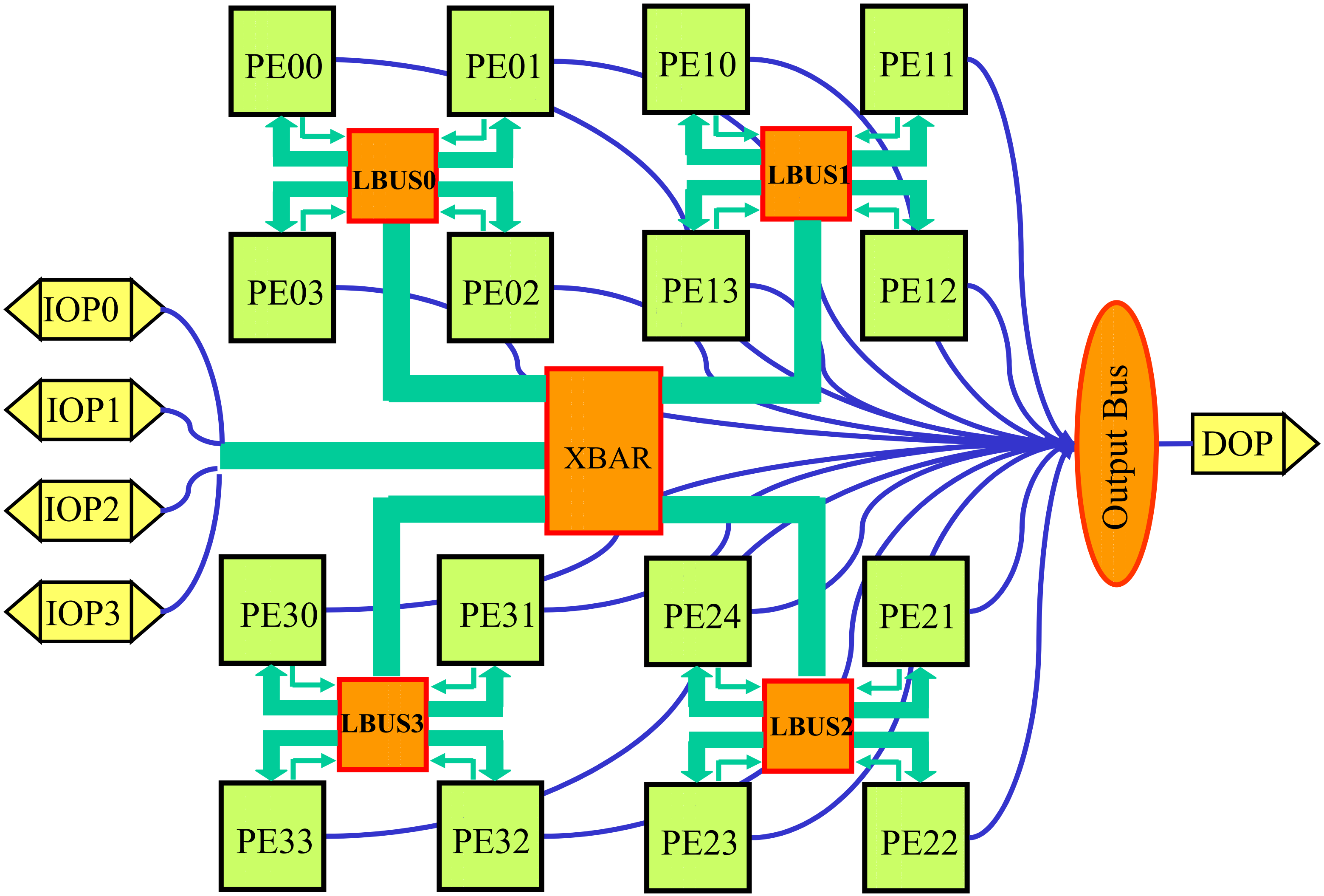
Power- and Space-efficient

A fraction of the overhead of FPGAs

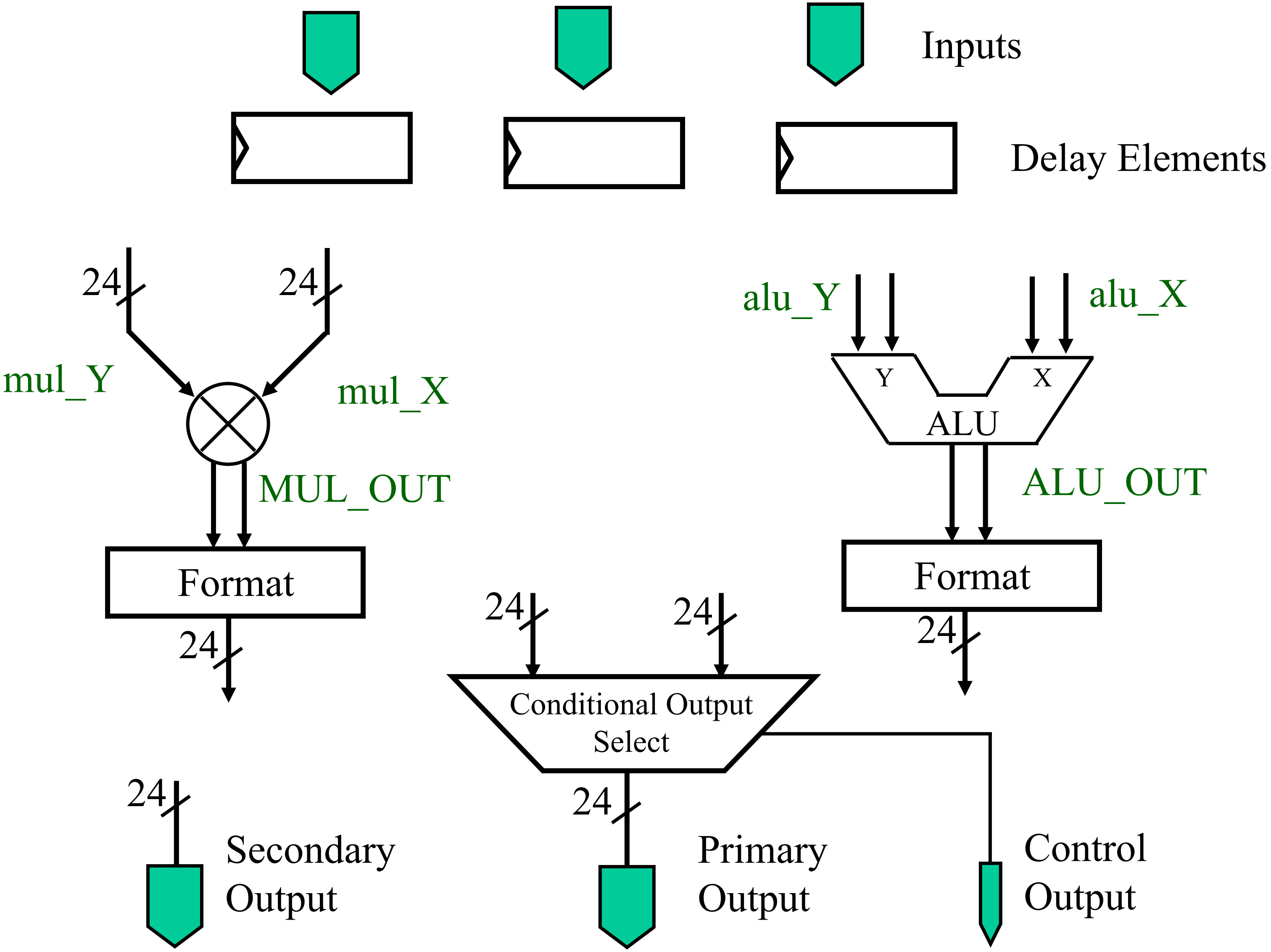
FPGA: < 1% area utilization for logic

RDPP: > 80% area utilization for logic

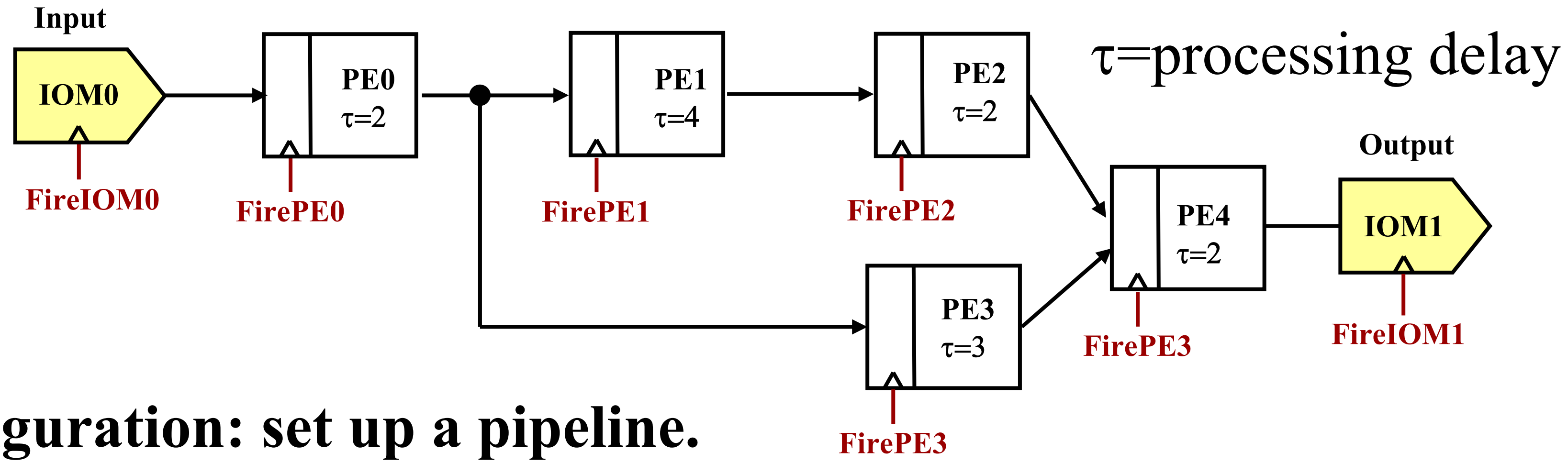
RDPP Data Path Elements



Processing Element Components



RDPP Pipeline Execution



Configuration: set up a pipeline.

1. Configure elements and interconnects.
2. Load execution program.

Execution: run a pipeline

1. Initialize
2. Loop
3. Terminate

Synchronous Pipeline

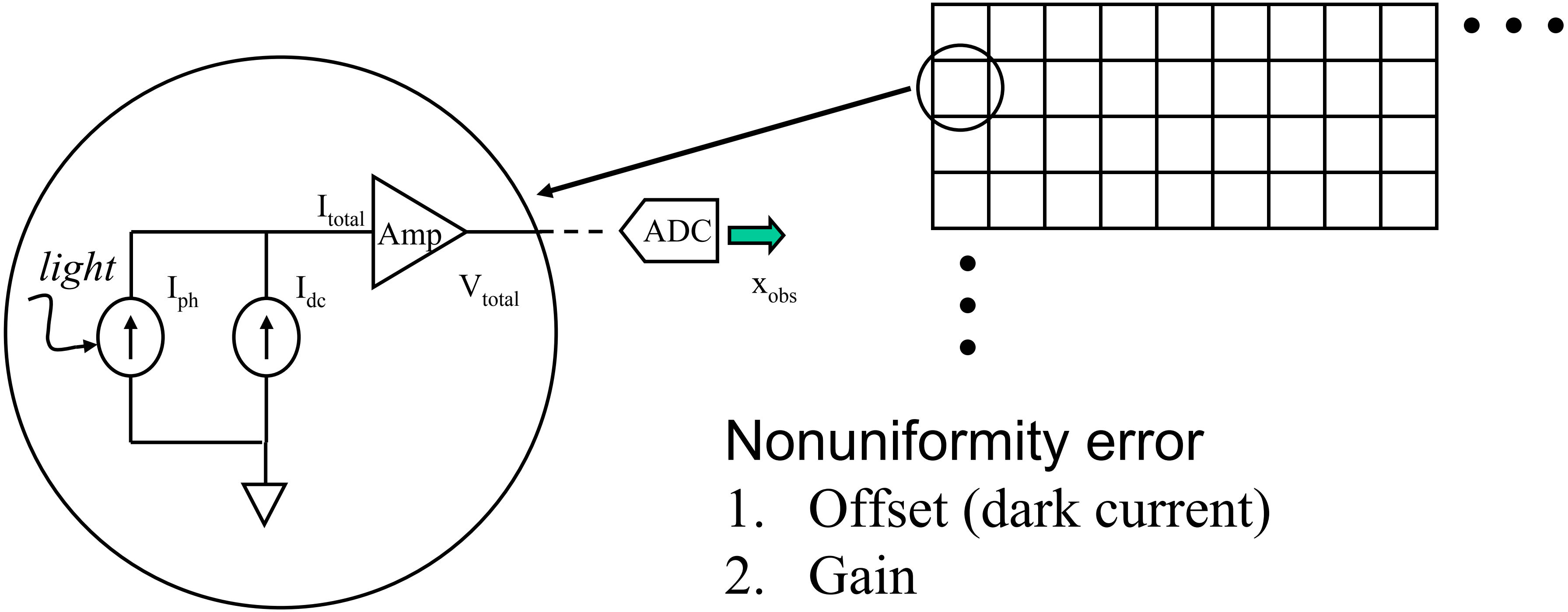
- *Blocking read*
- *Non-blocking write*
- *PEs and IO modules are dataflow actors*

Execute program until completed, or halted by internal or external events.

Focal Plane Array Sensor Readout Correction

FPA Readout Errors

Solid State Focal Plane Imaging Array



Nonuniformity error

1. Offset (dark current)
2. Gain

Bad Pixels

1. Hot pixels
2. Dead pixels

FPA Readout Correction

The amount of error is different for each pixel. Each pixel must be calibrated separately.

Phase 1: Calibration

Acquire calibration data for each pixel

1. Offset
2. Gain error
3. Detect bad pixels

Phase 2: Correction

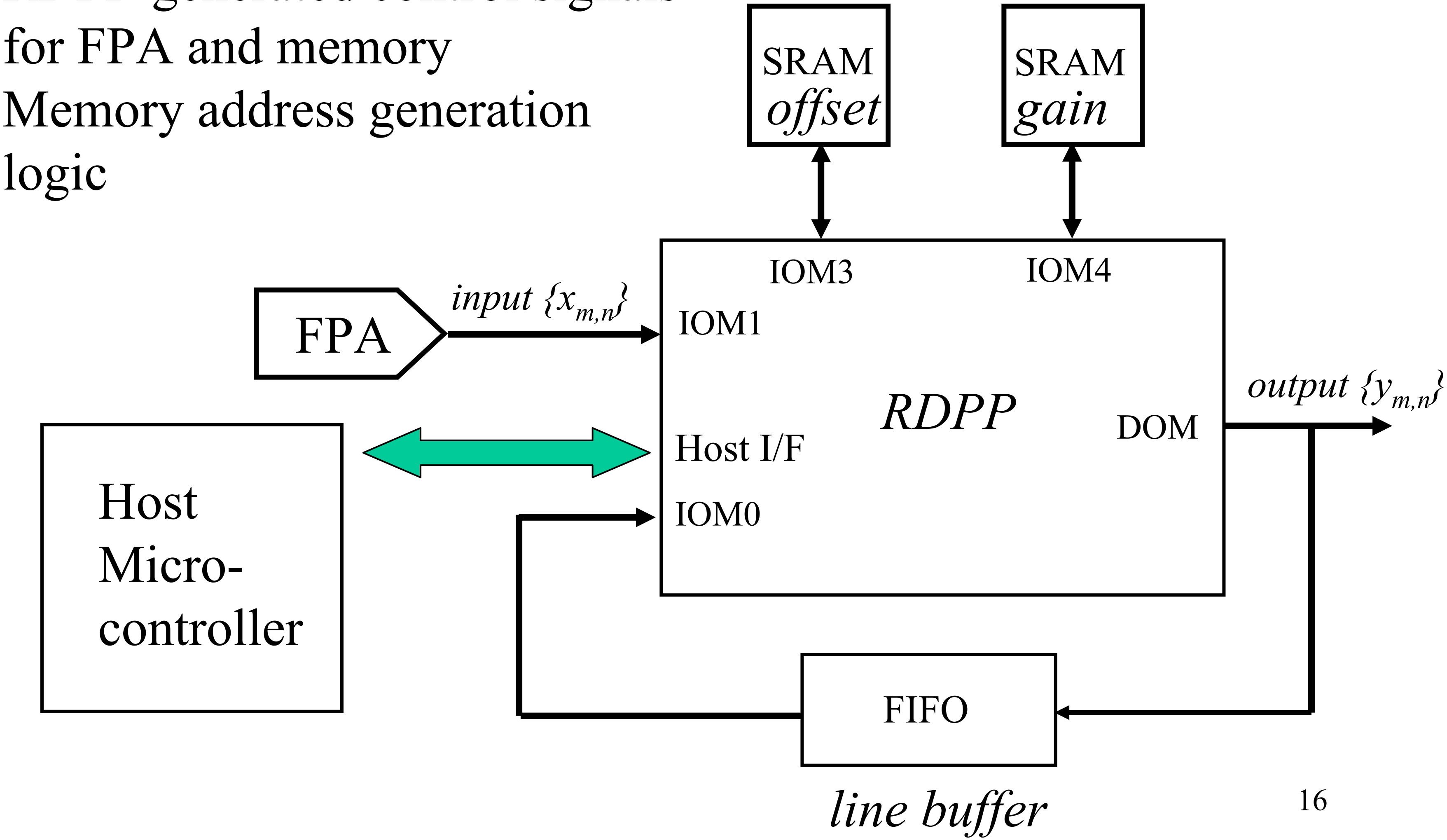
Read an image and correct each pixel

1. Multiply by 1/gain
2. Subtract offset
3. Replace bad pixels with a spatial average of neighbors

Hardware Configuration

Not shown:

1. RDPP-generated control signals for FPA and memory
2. Memory address generation logic



Phase 1: Calibration Data

Dark Current Offset Error

1. Close aperture
2. Capture image, store in RAM

Gain Error

1. Present known illumination
2. Capture image, store in RAM
3. Find “expected” value: image mean, compensated for offset.
4. Compute gain error from measured & expected values

Tag Bad Pixels

Pixels that don’t change in first two steps are “dead”

Use high-order bit of offset value to tag “dead” pixels

Reciprocal Gain Error

RDPP Has no hardware divider

Pre-compute reciprocal gain errors & store in memory

Store reciprocal in memory

Division Algorithm: Recurrence

$$a = \frac{g}{d} = \frac{g \cdot x^{(0)} \cdot x^{(1)} \cdot \ldots \cdot x^{(m-1)}}{d \cdot x^{(0)} \cdot x^{(1)} \cdot \ldots \cdot x^{(m-1)}} \qquad \left\{ \begin{array}{l} g\text{=gain error} \\ a\text{=reciprocal } (1/g) \\ d\text{=denominator} \end{array} \right.$$

Find $x^{(i)}$ such that numerator converges to 1

Set
$$d^{(0)} = d$$
$$x^{(i)} = 2 - d^{(i)}$$
$$d^{(i+1)} = d^{(i)} \cdot (2 - d^{(i)})$$

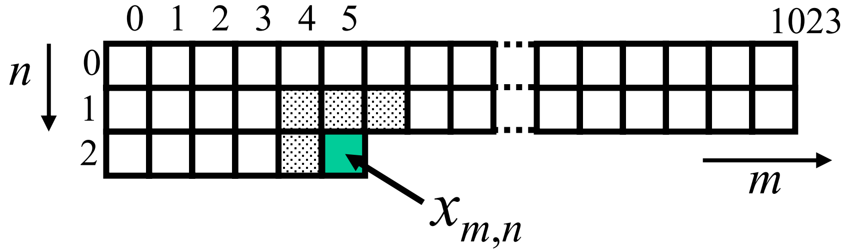
Easily done in a pipeline and/or tight loop in the RDPP

Phase 2: Correction

Nonuniformity Correction

$$x_{corr} = a \cdot (x_{obs} - x_{dc})$$

Bad Pixel Replacement



$$x_{m,n} = \frac{1}{4}\left(x_{m-1,n-1} + x_{m,n-1} + x_{m+1,n-1} + x_{m-1,n}\right)$$

¼ = two right shifts

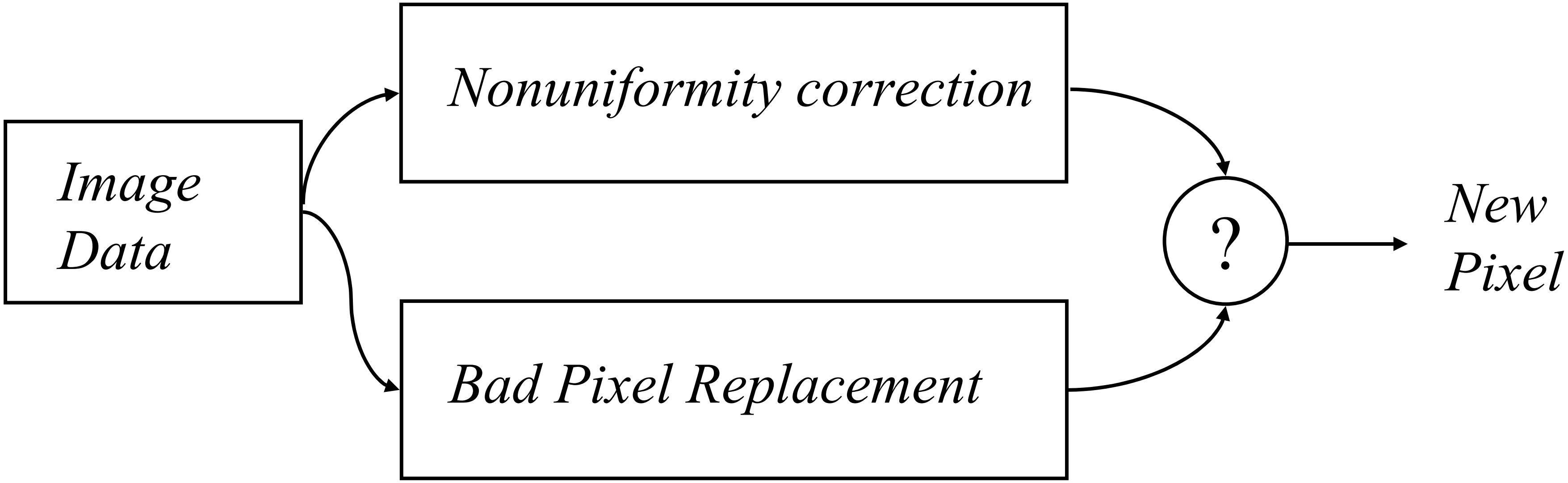
This is a “causal” solution; minimizes line buffers

Dynamic Data Switching

RDPP Does not have conditional branching

Instead, use conditional data path switching

Every PE can be a switching element

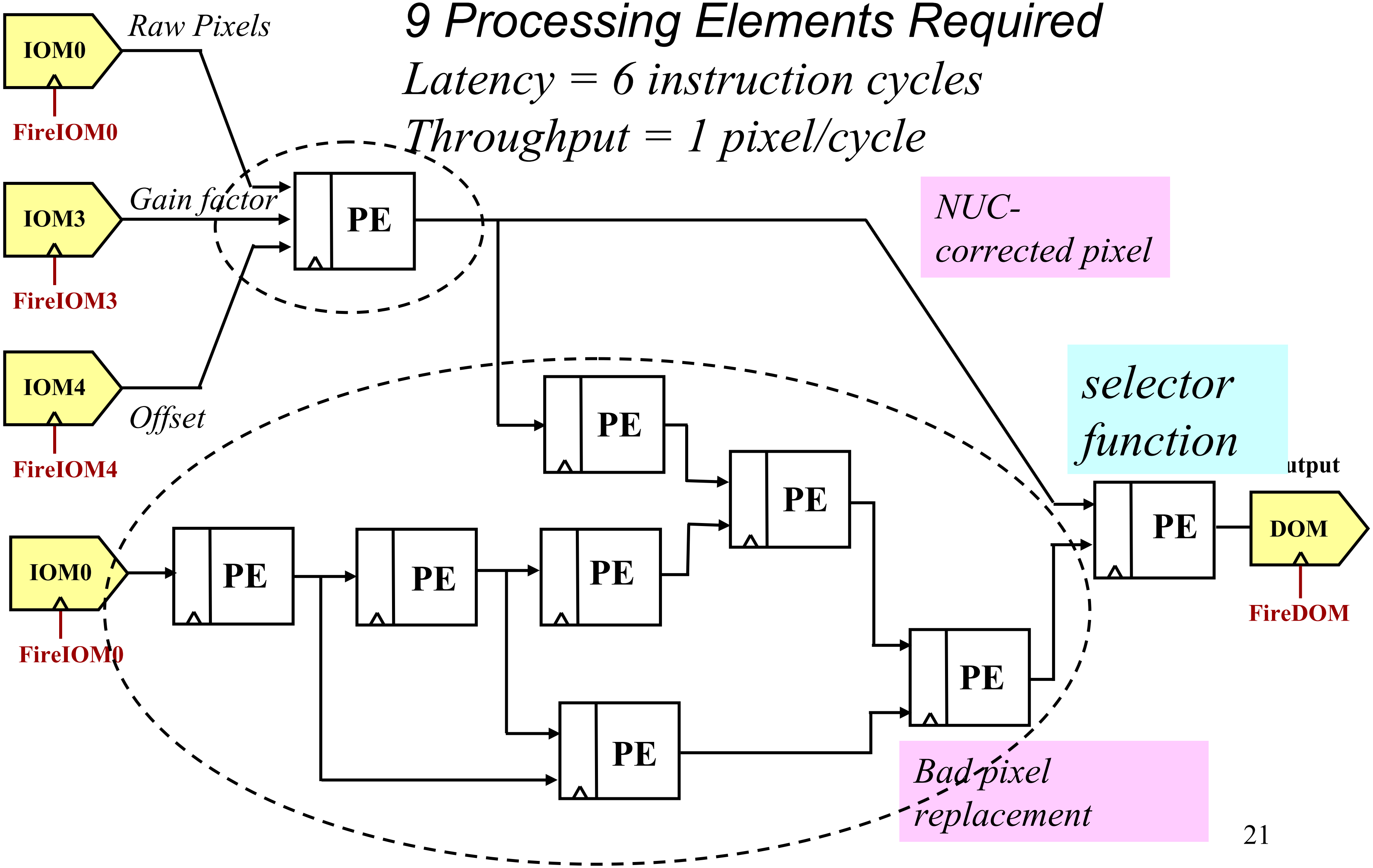


Correction Pipeline

9 Processing Elements Required

Latency = 6 instruction cycles

Throughput = 1 pixel/cycle



IOM0

Raw Pixels

FireIOM0

IOM3

Gain factor

FireIOM3

IOM4

Offset

FireIOM4

PE

NUC-corrected pixel

selector function

utput

DOM

FireDOM

IOM0

FireIOM0

PE

PE

PE

PE

PE

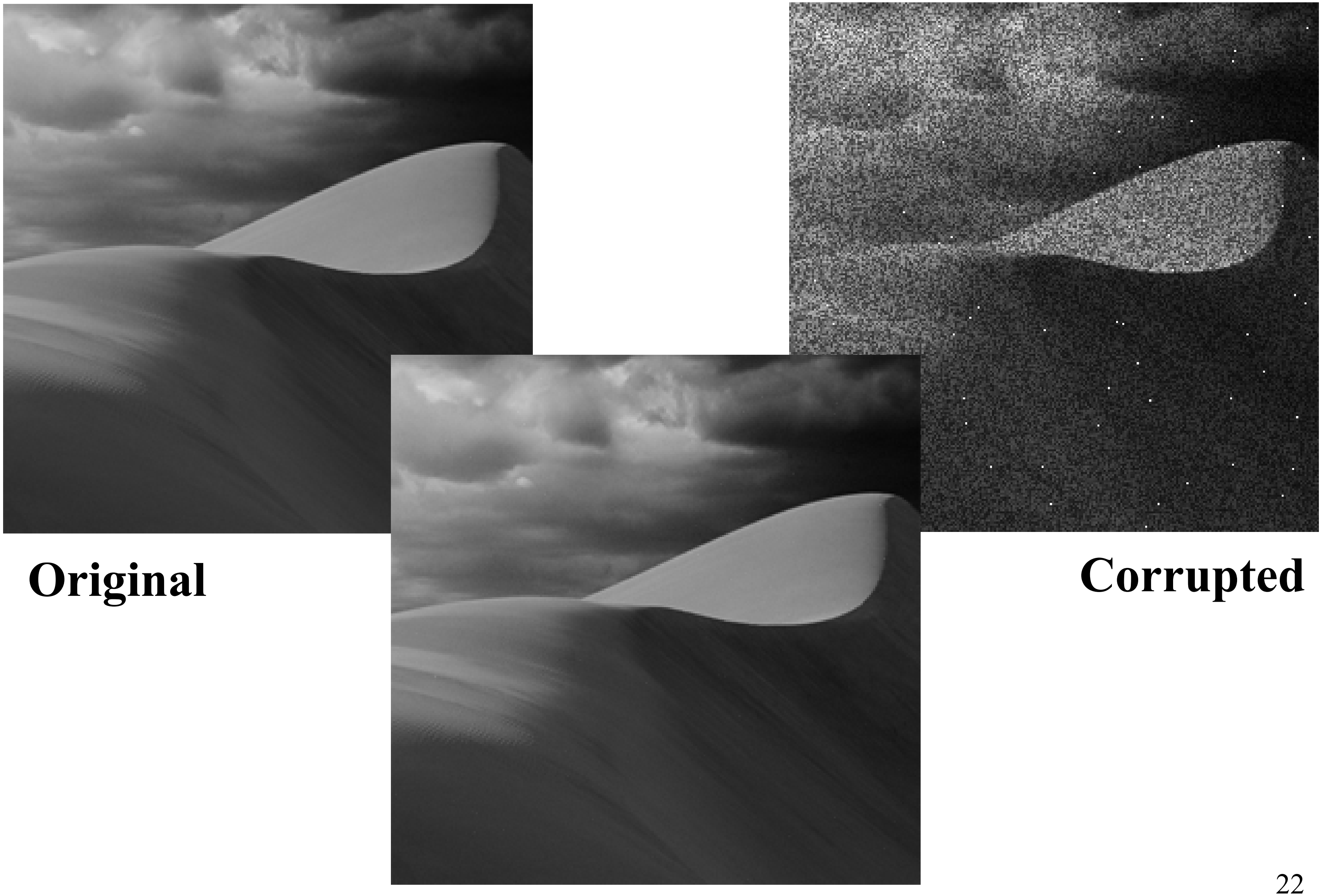
PE

PE

PE

Bad pixel replacement

RDPP Simulation Results



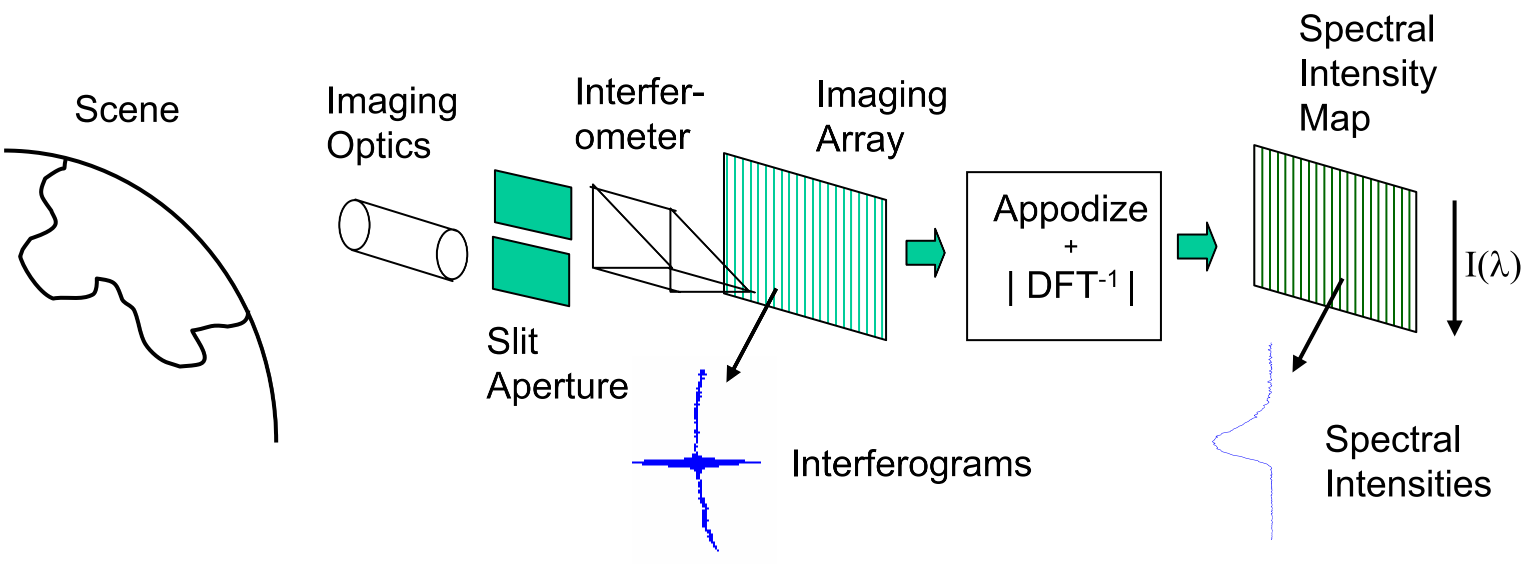
Original

Corrupted

Corrected

Fourier Transform Hyperspectral Imager Data Conversion

FTHSI System



Vertical columns on imaging array contain the Fourier Transform of the spectral intensity.

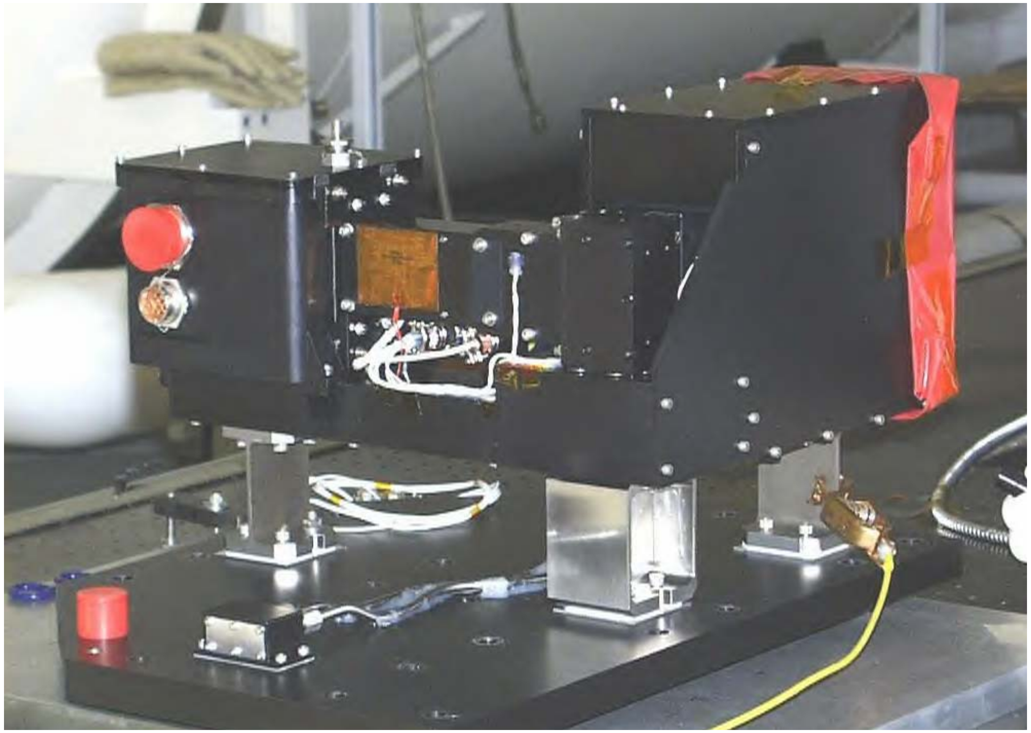
We must

1. **clean up the signal (appodize)**
2. **take the magnitude inverse Fourier transform.**

Kestrel FTHSI

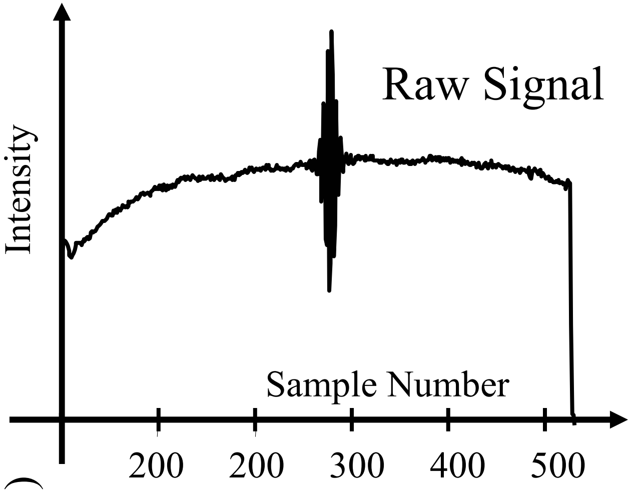
Flown on Air Force MightySat II.1 Satellite

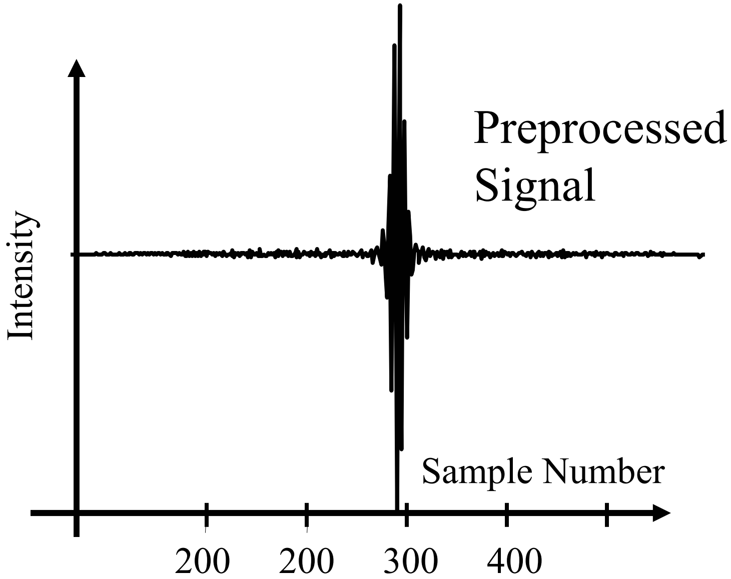
- Launched July 19, 2000
- AFRL Space Vehicles Directorate proof-of-concept "lab bench"

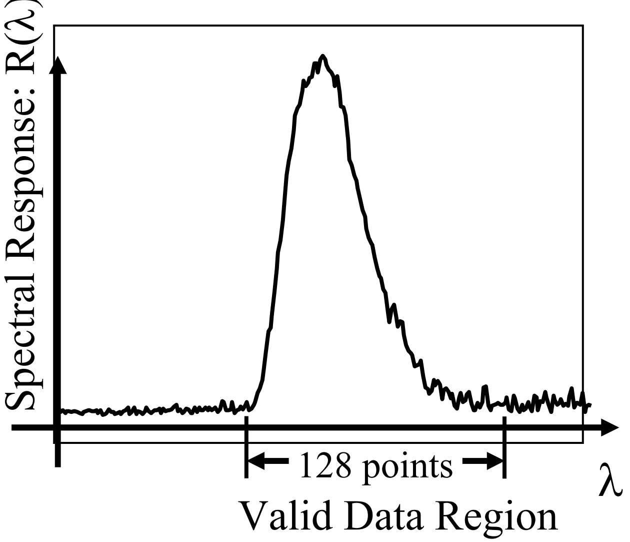


- Built by Kestrel Corporation.
- 470-1050 nm spectral range
- One picture every 3 days
- On-board data conversion experiment did not function.
- On the ground: **15 minutes** to convert an image

Kestrel "HyperCam" Data





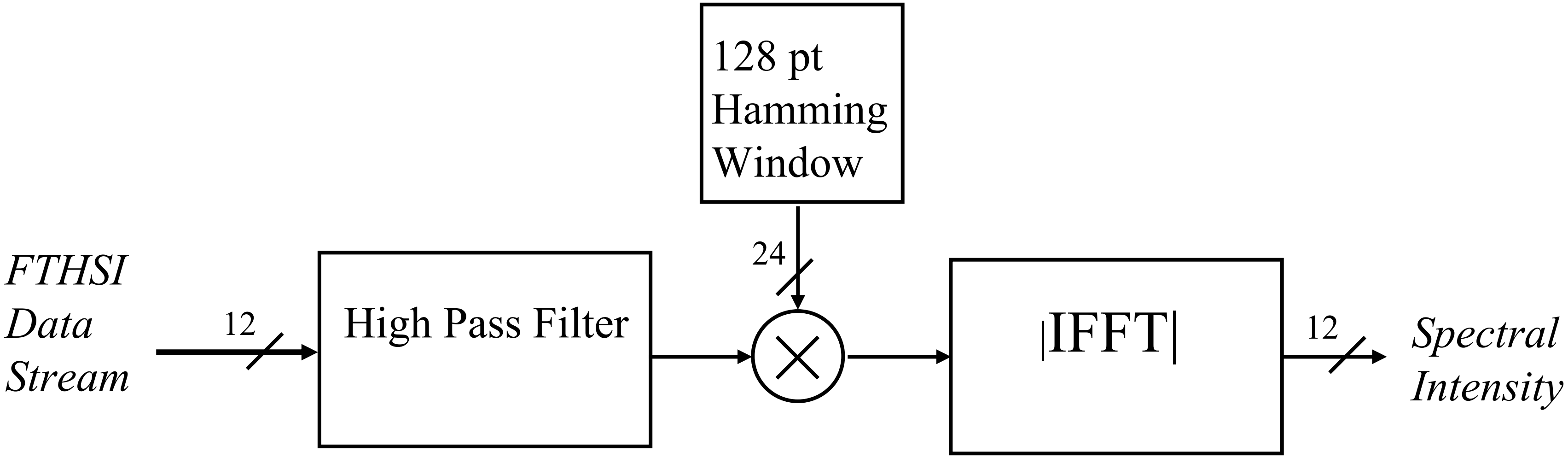


Converted signal

- Appodized, zero-mean
- Hamming window
- Inverse FFT
- Cropped

Data Courtesy of Dr. Leonard J. Otten III
Kestrel Corporation

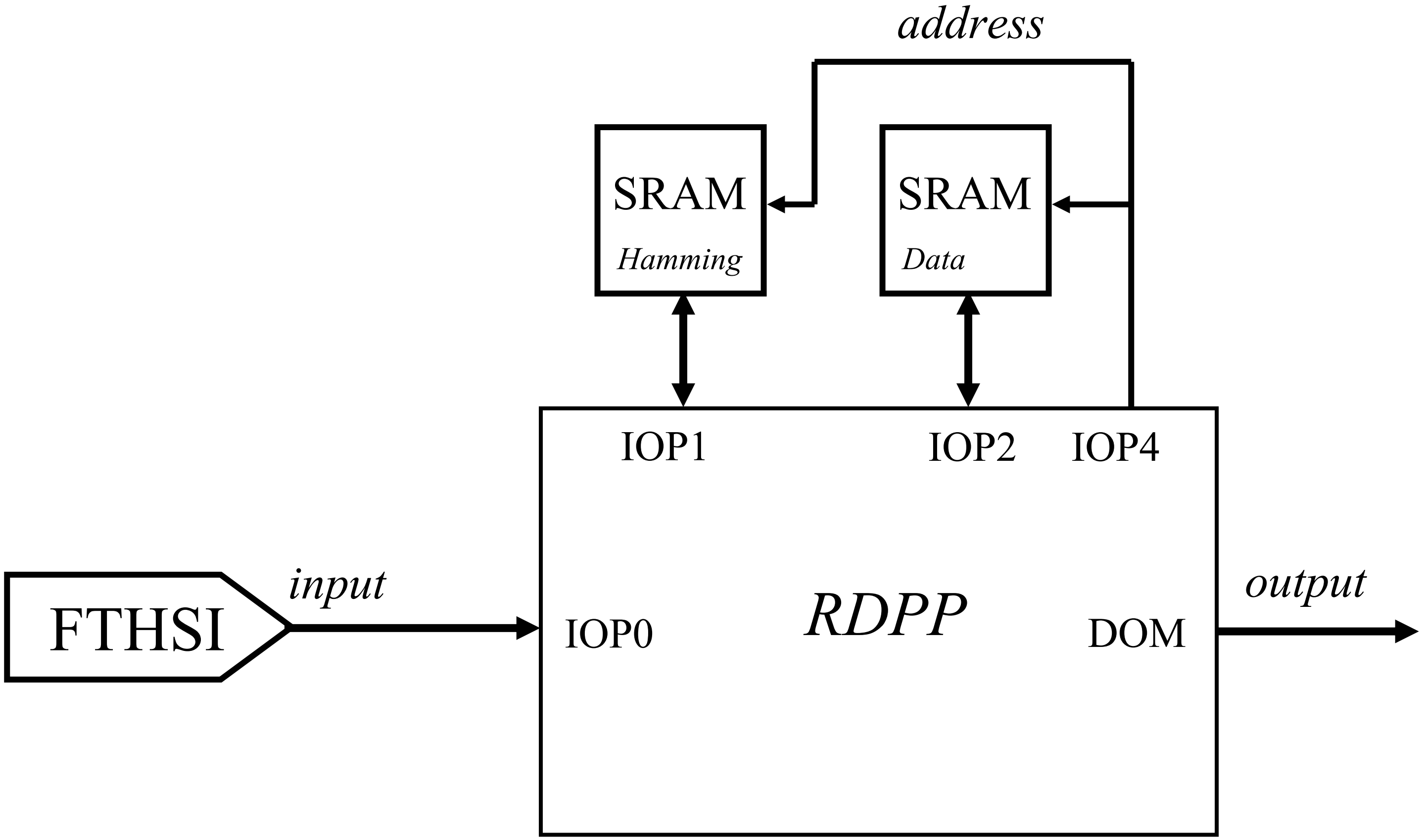
FTHSI Data Flow



Raw interferograms
128 points/column

Appodization requires 1 RDPP + 1 memory.
FFT computed with 1 to 8 RDPP chips.

FTHSI Hardware Configuratior



FFT Algorithms Compared

Cooley-Tukey

Optimized for single-multiplier machine

Minimizes number of multiply operations

Requires complex data shuffling

Multiple passes through the data

Goertzel

Requires more multiplies

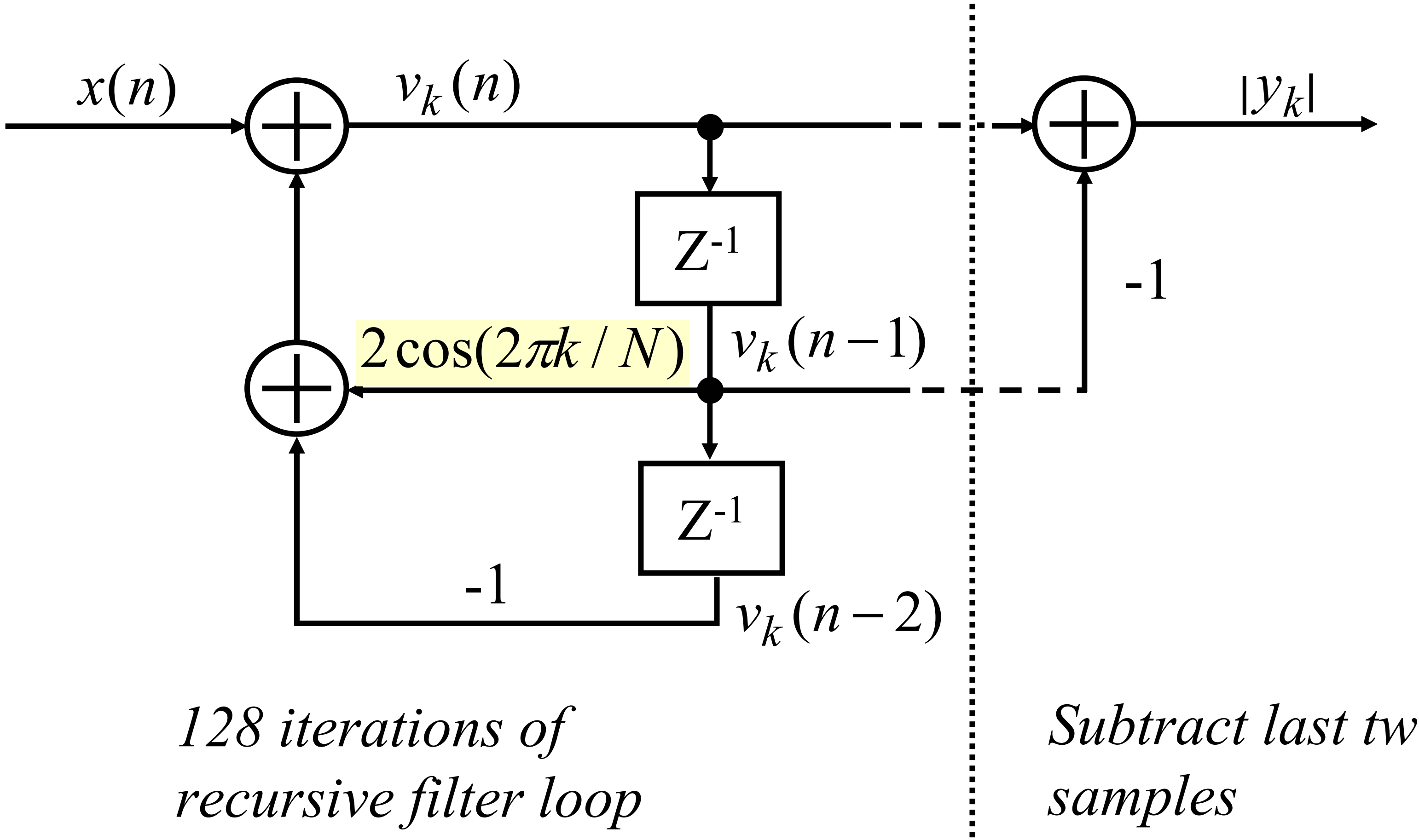
Fourier coefficents computed “in place”

Not data shuffling

Easily parallelized

RDPP is a multiplier-rich, parallel data path environment, well-suited to the Goertzel algorithm.

Compute One FFT Coefficient



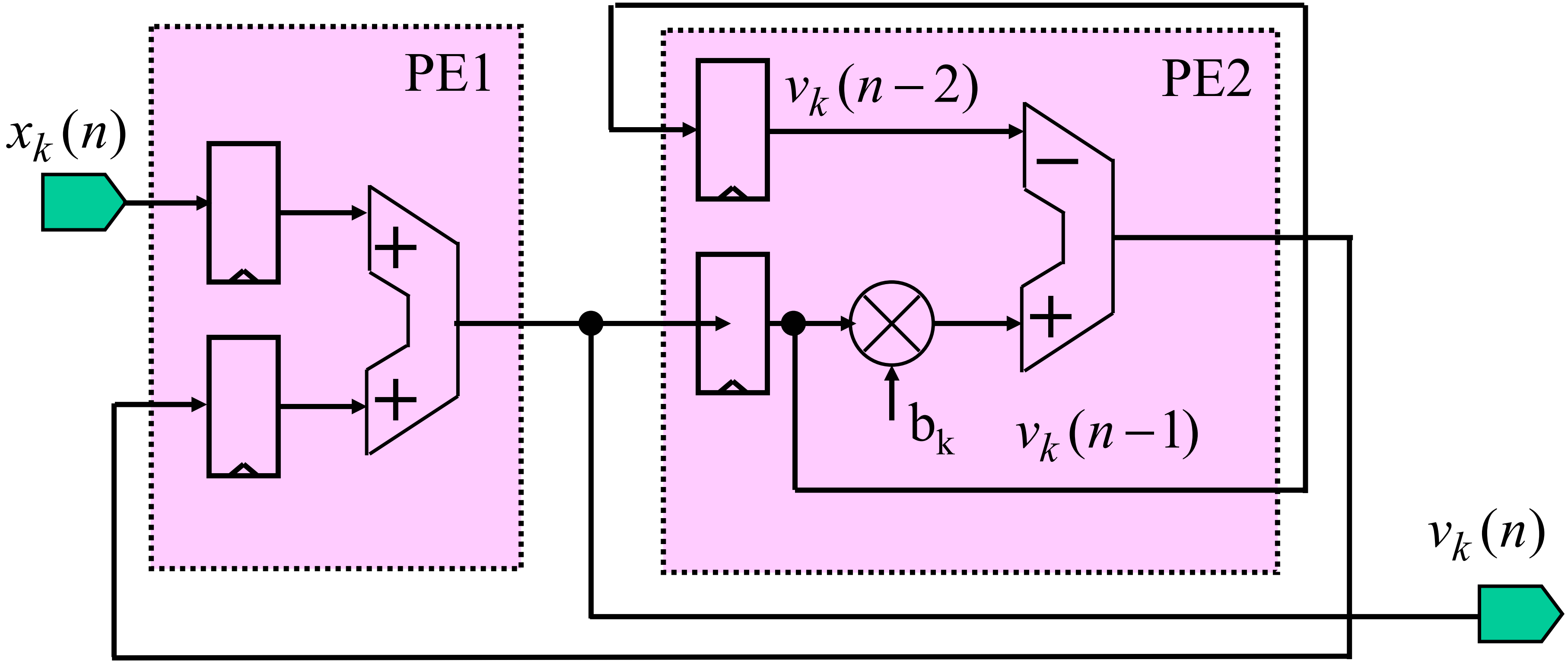
128 iterations of recursive filter loop

Subtract last two samples

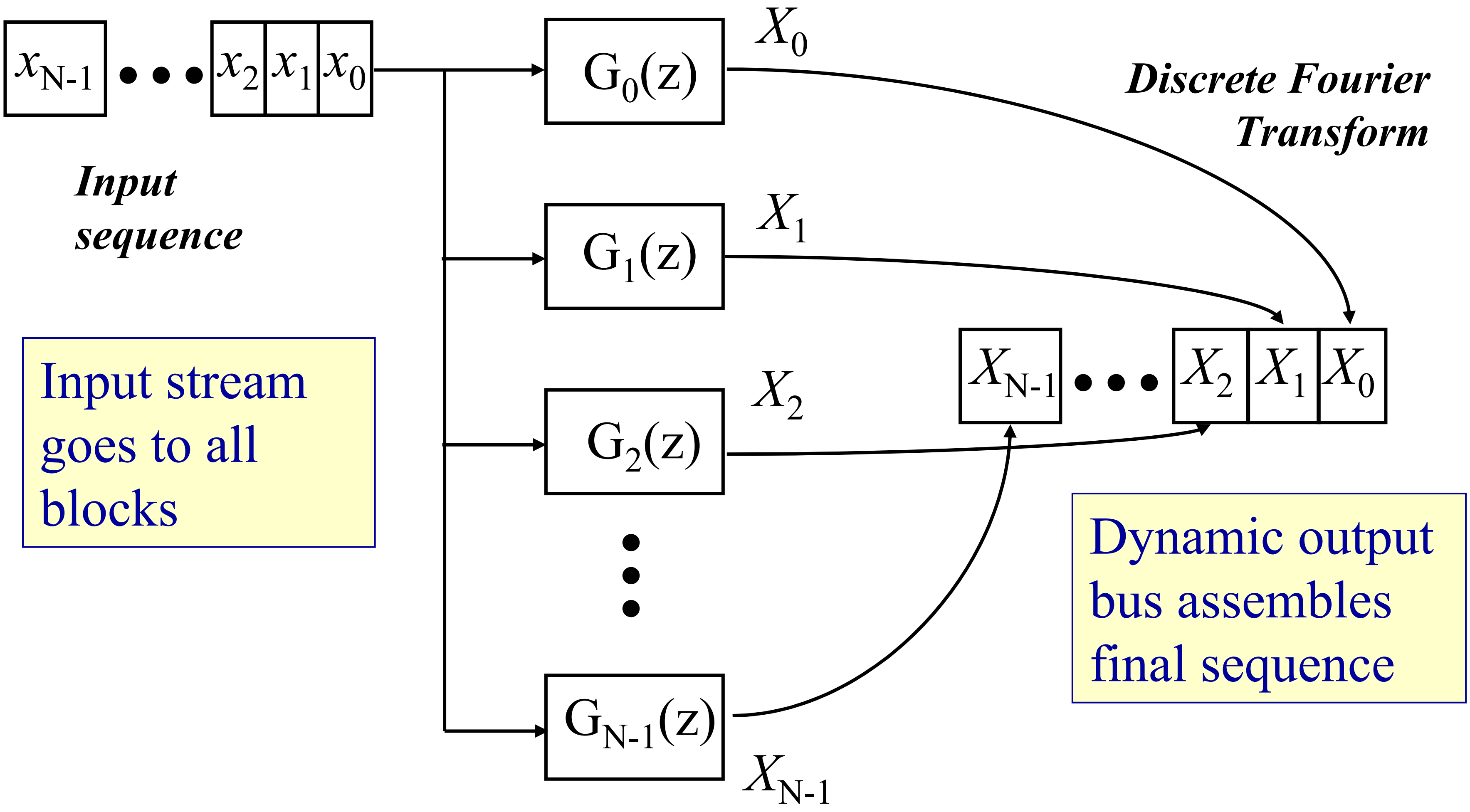
Coefficients computed in place

Identical except for $2\cos(2\pi k/N)$ factor

Two PEs Per DFT Value



Bank of Goertzel Filters


x_{N-1} • • • x_2 x_1 x_0
Input sequence
$G_0(z)$
X_0
Discrete Fourier Transform
$G_1(z)$
X_1
$G_2(z)$
X_2
X_{N-1} • • • X_2 X_1 X_0
$G_{N-1}(z)$
X_{N-1}

Input stream goes to all blocks

Dynamic output bus assembles final sequence

Only one read through the data required.
Can directly compute magnitude only, if desired.

Performance

One-Chip Solution

One RDPP chip can compute 8 DFT coefficients

For 128 points, compute in 8-coefficient “chunks”

8 passes through the data

Must re-load 8 coefficient registers per pass

Computation Time

Target instruction cycle rate: 60 MHz

128 samples × 2.133 μsec/sample × 8 = 2.18 msec

Configuration Time (Worst Case)

Reload 8 24-bit data registers, 8 times

27 clock cycles/register

27 cycles × 8 registers × 8 × 2.133 μsec/cycle = 3.69 msec

Total: 5.87 msec/conversion or 170 conversions/sec

Estimated Performance

Multi-RDPP Solution

8 RDPP chips for FFT + 1 RDPP for appodization
Compute FFT coefficients in parallel
Only one configuration

Computation

128 samples × 2.133 msec/sample = 0.273 msec/conversion
3,660 conversions/sec
7.15 sec to convert an image of
512 pixels
128 spectral bands,

Compare to Air Force MightySat FTHSI Experiments

~126 times faster than ground-based conversion with workstation.

Conclusions

Implemented both challenge problems in RDPPSim simulator

Both yield correct results

Validates the RDPP architecture for these applications

Validates RDPP development software

RDPP Features Demonstrated

Reconfigurability

Highly parallel data path processing

Dynamic data path switching

Partition large problem with incremental reconfiguration

Scalability by tiling RDPP chips